# x86-64 Reference Sheet (GNU assembler format)

Simone Goggiani

## Instructions

### Data movement

| | |
|---|---|
| `movq` Src, Dest | Dest = Src |
| `movsbq` Src,Dest | Dest (quad) = Src (byte), sign-extend |
| `movzbq` Src,Dest | Dest (quad) = Src (byte), zero-extend |

### Conditional move

| | |
|---|---|
| `cmove` Src, Dest | Equal / zero |
| `cmovne` Src, Dest | Not equal / not zero |
| `cmovs` Src, Dest | Negative |
| `cmovns` Src, Dest | Nonnegative |
| `cmovg` Src, Dest | Greater (signed >) |
| `cmovge` Src, Dest | Greater or equal (signed ≥) |
| `cmovl` Src, Dest | Less (signed <) |
| `cmovle` Src, Dest | Less or equal (signed ≤) |
| `cmova` Src, Dest | Above (unsigned >) |
| `cmovae` Src, Dest | Above or equal (unsigned ≥) |
| `cmovb` Src, Dest | Below (unsigned <) |
| `cmovbe` Src, Dest | Below or equal (unsigned ≤) |

### Control transfer

| | |
|---|---|
| `cmpq` Src2, Src1 | Sets CCs Src1−Src2 |
| `testq` Src2, Src1 | Sets CCs Src1 & Src2 |
| `jmp label` | jump (always) |
| `je label` | jump equal |
| `jne label` | jump not equal |
| `js label` | jump negative |
| `jns label` | jump non-negative |
| `jg label` | jump greater (signed >) |
| `jge label` | jump greater or equal (signed ≥) |
| `jl label` | jump less (signed <) |
| `jle label` | jump less or equal (signed ≤) |
| `ja label` | jump above (unsigned >) |
| `jb label` | jump below (unsigned <) |
| `pushq` Src | %rsp = %rsp − 8, Mem[%rsp] = Src |
| `popq` Dest | Dest = Mem[%rsp], %rsp = %rsp + 8 |
| `call label` | push address of next instruction, `jmp label` |
| `ret` | %rip = Mem[%rsp], %rsp = %rsp + 8 |

`setx` set single byte based on combination of CCs

## Arithmetic operations

| | |
|---|---|
| `leaq` Src, Dest | Dest = address of Src |
| `incq` Dest | Dest = Dest + 1 |
| `decq` Dest | Dest = Dest − 1 |
| `addq` Src, Dest | Dest = Dest + Src |
| `subq` Src, Dest | Dest = Dest − Src |
| `imulq` Src, Dest | Dest = Dest * Src |
| `xorq` Src, Dest | Dest = Dest ^ Src |
| `orq` Src, Dest | Dest = Dest \| Src |
| `andq` Src, Dest | Dest = Dest & Src |
| `negq` Dest | Dest = − Dest |
| `notq` Dest | Dest = ~ Dest |
| `salq` *k*, Dest | Dest = Dest ≪ *k* |
| `sarq` *k*, Dest | Dest = Dest ≫ *k* (arithmetic) |
| `shrq` *k*, Dest | Dest = Dest ≫ *k* (logical) |

## Addressing modes

- **Immediate**
  $val Val
  val: constant integer value
  `movq $7, %rax`
- **Normal**
  (R)=Mem[Reg[R]]
  R: register R specifies memory address
  `movq (%rcx), %rax`
- **Displacement**
  D(R)=Mem[Reg[R]+D]
  R: register specifies start of memory region
  D: constant displacement D specifies offset
  `movq 8(%rdi), %rdx`
- **Indexed**
  D(Rb,Ri,S) Mem[Reg[Rb]+S*Reg[Ri]+D]
  D: constant displacement 1, 2, or 4 bytes
  Rb: base register: any of 8 integer registers
  Ri: index register: any, except %esp
  S: scale: 1, 2, 4, or 8
  `movq 0x100(%rcx,%rax,4), %rdx`

## Instruction suffixes

| | | |
|---|---|---|
| `b` | byte | 8 |
| `w` | word (2 bytes) | 16 |
| `l` | long (4 bytes) | 32 |
| `q` | quad (8 bytes) | 64 |

## Condition codes

| | | |
|---|---|---|
| **CF** | Carry Flag | (U) |
| **ZF** | Zero Flag | == 0 |
| **SF** | Sign Flag | < 0 |
| **OF** | Overflow Flag | (T) |

## Integer registers

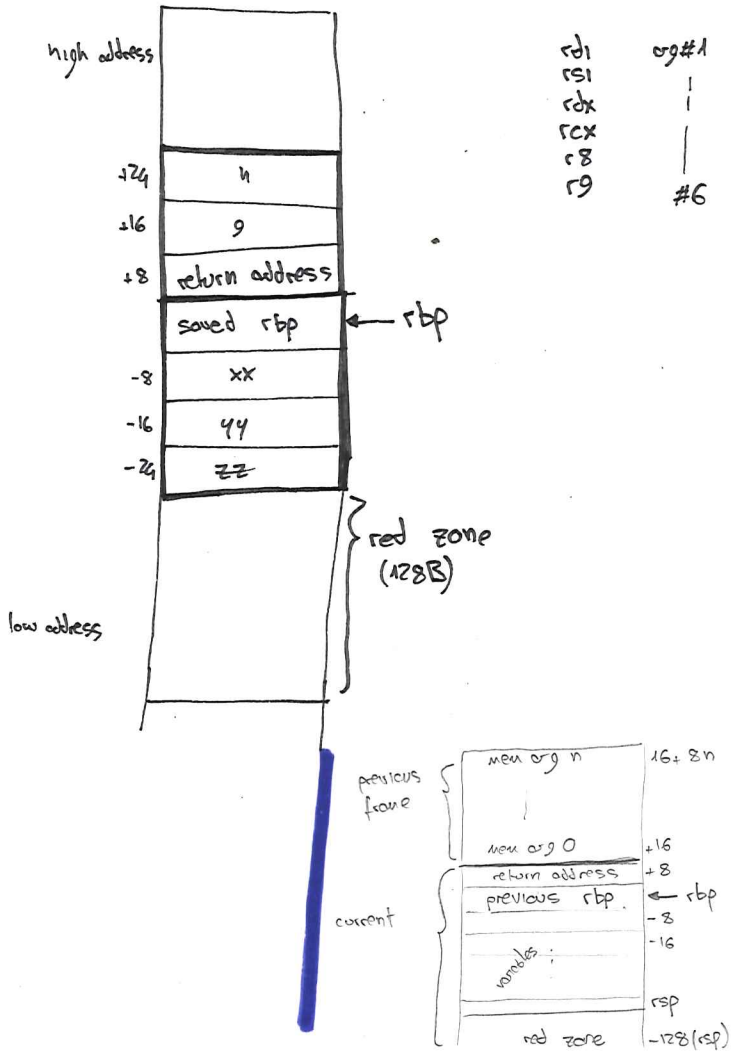
| | |
|---|---|
| %rax | Return value |
| %rbx | Callee saved |
| %rcx | 4th argument |
| %rdx | 3rd argument |
| %rsi | 2nd argument |
| %rdi | 1st argument |
| %rbp | Callee saved |
| %rsp | Stack pointer |
| %r8 | 5th argument |
| %r9 | 6th argument |
| %r10 | Scratch register |
| %r11 | Scratch register |
| %r12 | Callee saved |
| %r13 | Callee saved |
| %r14 | Callee saved |
| %r15 | Callee saved |

| 64 (really ext) | 32 (extended) | 16 | 2x 8 | |
|---|---|---|---|---|
| rax | eax | ax | (ah + al) | accumulate — return value |
| rbx | ebx | bx | (bh + bl) | base — callee saved |
| rcx | ecx | cx | (ch + cl) | counter — arg#4 |
| rdx | edx | dx | (dh + dl) | data — arg#3 — 2nd return |
| rsi | esi | si | | source index — arg#2 |
| rdi | edi | di | | dest index — arg#1 |
| rsp | esp | sp | | stack ptr |
| rbp | ebp | bp | | base ptr — callee saved - frame ptr |
| rip | eip | ip | | inst. ptr |
| rsr | esr | sr | | status (flags) |
| r8 | r8d | | | numbered — 8 arg#5, 9 arg#6, 10 static chain ptr, 11 used for linking, 12–14 callee saved |
| : | : | | | |
| r15 | r15d | | | |

general purpose

## Conditions

| | | | |
|---|---|---|---|
| e | ZF | ne | ~ZF |
| s | SF | ns | ~SF |
| g | ~(SF^OF)&~ZF | ge | ~(SF^OF) |
| l | (SF^OF) | le | (SF^OF)\|ZF |
| a | ~CF&~ZF | ae | |
| b | CF | be | |



# x86-64 Reference Sheet (GNU assembler format)

SIMONE GUGGIARI
ETH ZÜRICH 2016
LEGI # 14-932-867

## Instructions

### Data movement

| | |
|---|---|
| `movq` Src, Dest | Dest = Src |
| `movsbq` Src,Dest | Dest (quad) = Src (byte), sign-extend |
| `movzbq` Src,Dest | Dest (quad) = Src (byte), zero-extend |

### Conditional move

| | |
|---|---|
| `cmove` Src, Dest | Equal / zero |
| `cmovne` Src, Dest | Not equal / not zero |
| `cmovs` Src, Dest | Negative |
| `cmovns` Src, Dest | Nonnegative |
| `cmovg` Src, Dest | Greater (signed $>$) |
| `cmovge` Src, Dest | Greater or equal (signed $\geq$) |
| `cmovl` Src, Dest | Less (signed $<$) |
| `cmovle` Src, Dest | Less or equal (signed $\leq$) |
| `cmova` Src, Dest | Above (unsigned $>$) |
| `cmovae` Src, Dest | Above or equal (unsigned $\geq$) |
| `cmovb` Src, Dest | Below (unsigned $<$) |
| `cmovbe` Src, Dest | Below or equal (unsigned $\leq$) |

### Control transfer

| | |
|---|---|
| `cmpq` Src2, Src1 | Sets CCs Src1 − Src2 |
| `testq` Src2, Src1 | Sets CCs Src1 & Src2 |
| `jmp label` | jump |
| `je label` | jump equal |
| `jne label` | jump not equal |
| `js label` | jump negative |
| `jns label` | jump non-negative |
| `jg label` | jump greater (signed $>$) |
| `jge label` | jump greater or equal (signed $\geq$) |
| `jl label` | jump less (signed $<$) |
| `jle label` | jump less or equal (signed $\leq$) |
| `ja label` | jump above (unsigned $>$) |
| `jb label` | jump below (unsigned $<$) |
| `pushq` Src | %rsp = %rsp − 8, Mem[%rsp] = Src |
| `popq` Dest | Dest = Mem[%rsp], %rsp = %rsp + 8 |
| `call label` | push address of next instruction, `jmp label` |
| `ret` | %rip = Mem[%rsp], %rsp = %rsp + 8 |

### Arithmetic operations

| | |
|---|---|
| `leaq` Src, Dest | Dest = address of Src |
| `incq` Dest | Dest = Dest + 1 |
| `decq` Dest | Dest = Dest − 1 |
| `addq` Src, Dest | Dest = Dest + Src |
| `subq` Src, Dest | Dest = Dest − Src |
| `imulq` Src, Dest | Dest = Dest $*$ Src |
| `xorq` Src, Dest | Dest = Dest ^ Src |
| `orq` Src, Dest | Dest = Dest \| Src |
| `andq` Src, Dest | Dest = Dest & Src |
| `negq` Dest | Dest = − Dest |
| `notq` Dest | Dest = $\sim$ Dest |
| `salq` $k$, Dest | Dest = Dest $\ll k$ |
| `sarq` $k$, Dest | Dest = Dest $\gg k$ (arithmetic) |
| `shrq` $k$, Dest | Dest = Dest $\gg k$ (logical) |

## Addressing modes

- **Immediate**
  $val Val
  val: constant integer value
  `movq $7, %rax`
- **Normal**
  (R) Mem[Reg[R]]
  R: register R specifies memory address
  `movq (%rcx), %rax`
- **Displacement**
  D(R) Mem[Reg[R]+D]
  R: register specifies start of memory region
  D: constant displacement D specifies offset
  `movq 8(%rdi), %rdx`
- **Indexed**
  D(Rb,Ri,S) Mem[Reg[Rb]+S*Reg[Ri]+D]
  D: constant displacement 1, 2, or 4 bytes
  Rb: base register: any of 8 integer registers
  Ri: index register: any, except %esp
  S: scale: 1, 2, 4, or 8
  `movq 0x100(%rcx,%rax,4), %rdx`

## Instruction suffixes

| | |
|---|---|
| `b` | byte |
| `w` | word (2 bytes) |
| `l` | long (4 bytes) |
| `q` | quad (8 bytes) |

## Condition codes

| | |
|---|---|
| **CF** | Carry Flag |
| **ZF** | Zero Flag |
| **SF** | Sign Flag |
| **OF** | Overflow Flag |

## Integer registers

| | |
|---|---|
| %rax | Return value |
| %rbx | Callee saved |
| %rcx | 4th argument |
| %rdx | 3rd argument |
| %rsi | 2nd argument |
| %rdi | 1st argument |
| %rbp | Callee saved |
| %rsp | Stack pointer |
| %r8 | 5th argument |
| %r9 | 6th argument |
| %r10 | Scratch register |
| %r11 | Scratch register |
| %r12 | Callee saved |
| %r13 | Callee saved |
| %r14 | Callee saved |
| %r15 | Callee saved |